The Effects of System Options on Code Performance

Courtenay T. Vaughan
ctvaugh@sandia.gov

Sandia National Laboratories
February 2007

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

XT3 Available Options

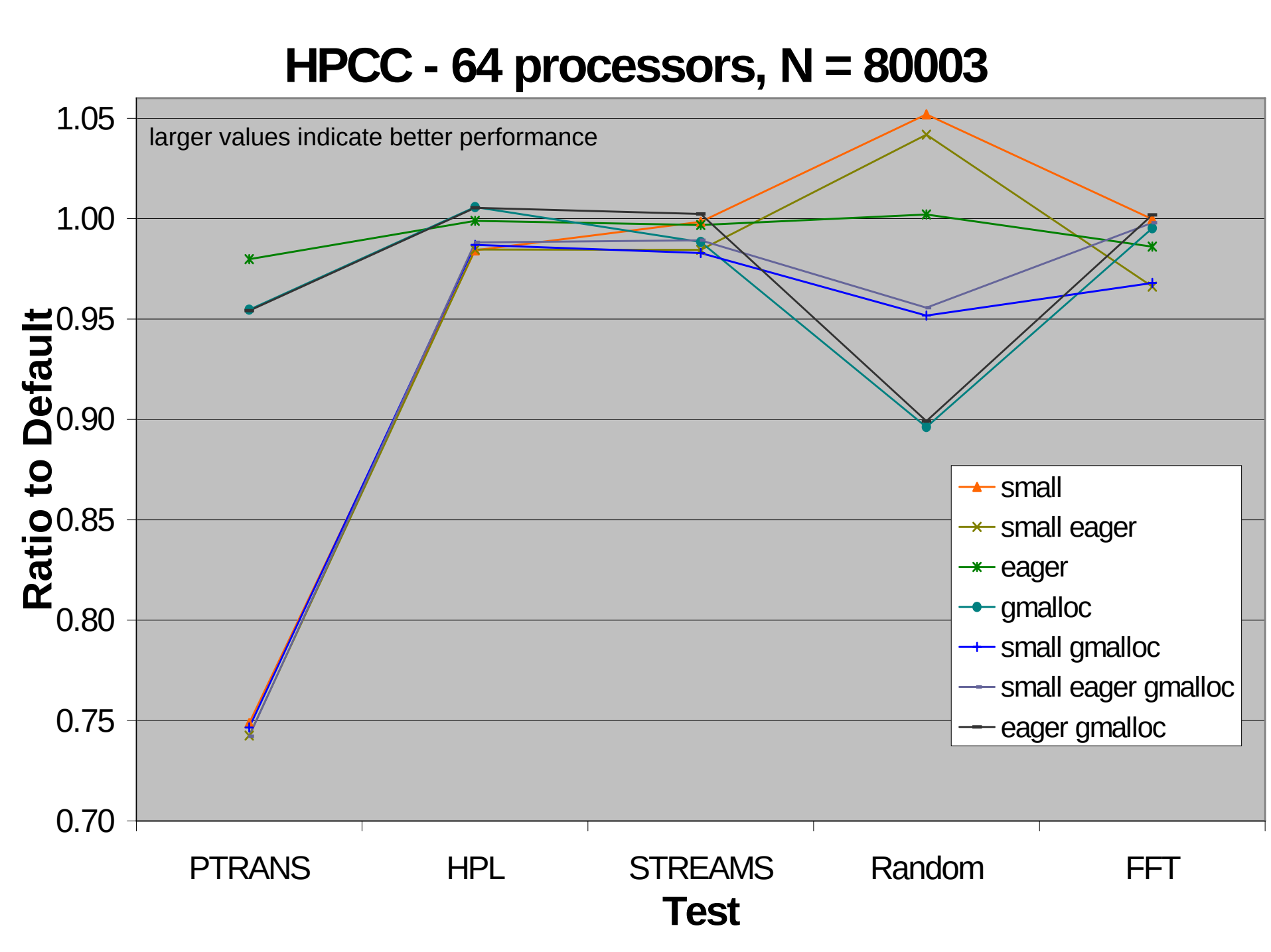
- small vs. large pages
  - Controls memory page size
  - large pages are default
  - yod -small_pages ...
- eager vs. rendezvous message sends
  - Controls message protocol
  - rendezvous protocol is default
  - export MPI_PTL_EAGER_LONG=1
- Catamount malloc vs. GNU malloc
  - Controls which malloc is being used
  - Link time option (cc ... -lgmalloc ...)

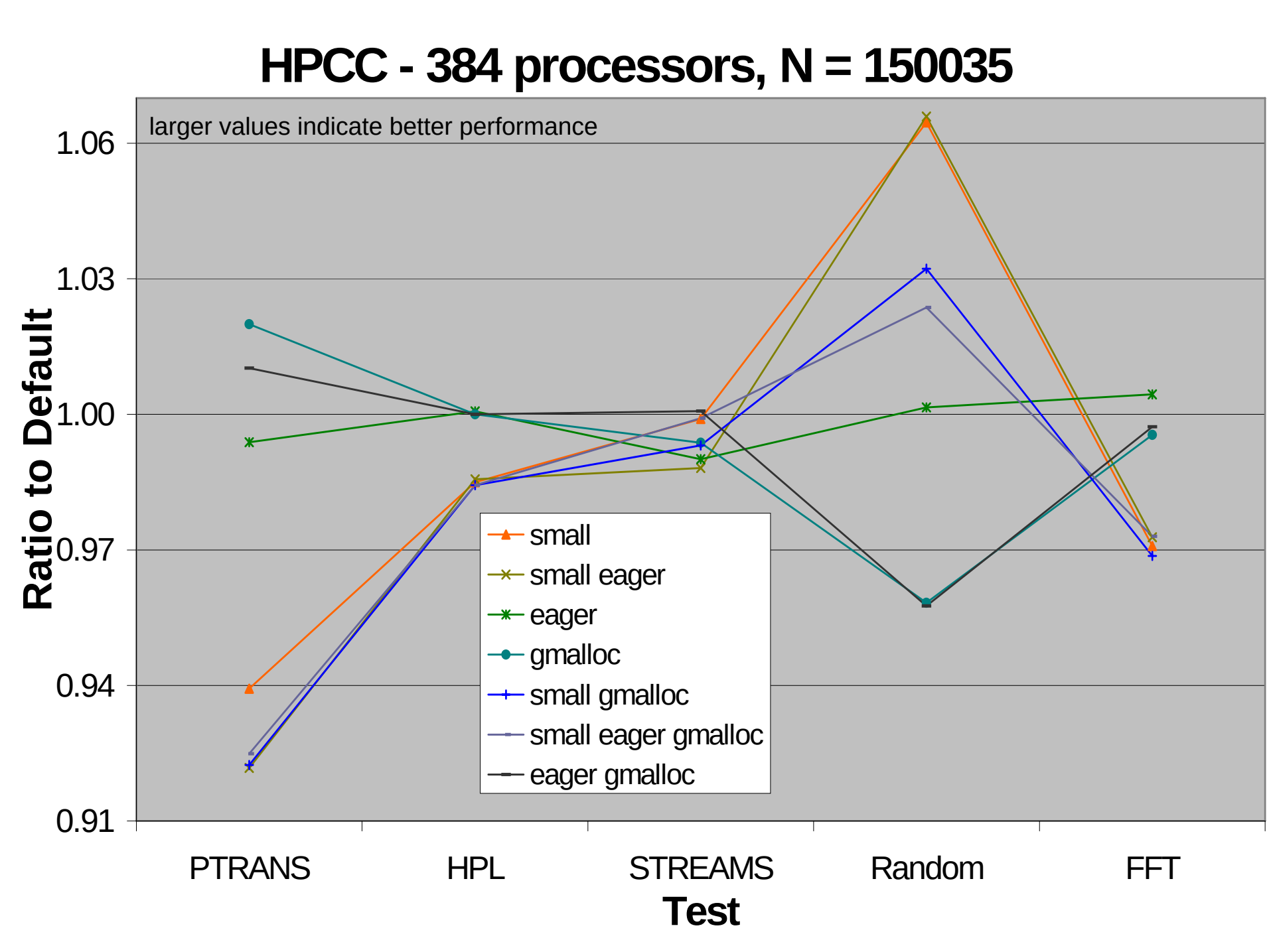
Test Parameters

- Codes were run for all combinations of options
- Codes were run using only one core per socket
- All results on a given number of processors for a given code were run using the same nodes on the machine
- Most of the results are from Red Storm (2.4 GHz processors) while some are from our test system (2.0 GHz processors)
- Most tests were run once
  - Early experience with the test system indicated that this was sufficient

HPCC

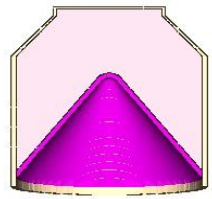
- Series of 7 benchmarks in one package. We are using 5 of them:
  - PTRANS - matrix transposition
  - HPL - Linpack direct dense system solve
  - STREAMS - Memory bandwidth
  - Random Access - Global random memory access
  - FFT - large 1-D FFT
- Code is C plus libraries


HPCC - 64 processors, N = 80003
larger values indicate better performance
Ratio to Default
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
PTRANS
HPL
STREAMS
Random
FFT
Test
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc

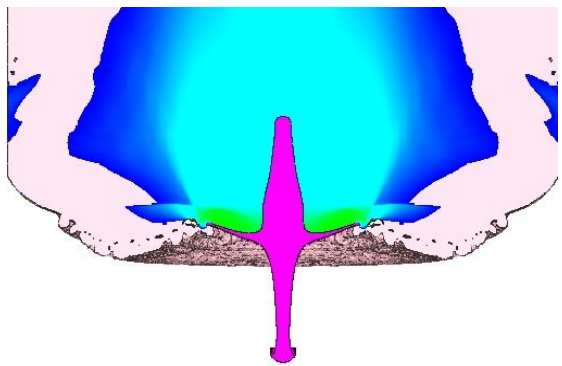

HPCC - 384 processors, N = 150035
larger values indicate better performance
Ratio to Default
1.06
1.03
1.00
0.97
0.94
0.91
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc
PTRANS
HPL
STREAMS
Random
FFT
Test

CTH

- **Three-dimensional shock hydrodynamics code**
- **Ran in flat mesh mode - no AMR (Automatic Mesh Refinement)**
- **Shaped charge problem**
- **90 x 216 x 90 cells per processor**
- **Code is mostly FORTRAN with a little C**

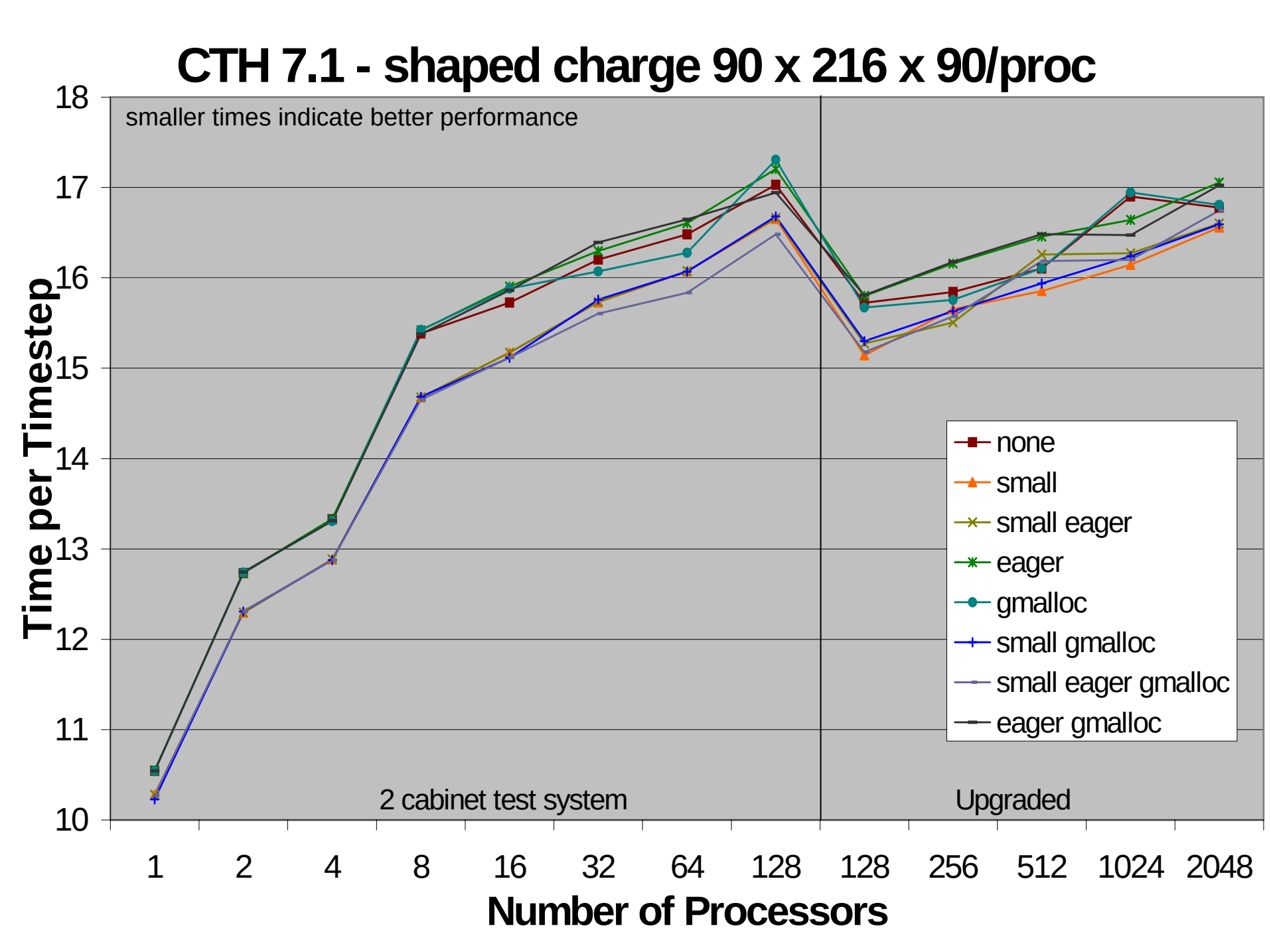


time = 0.0 ms



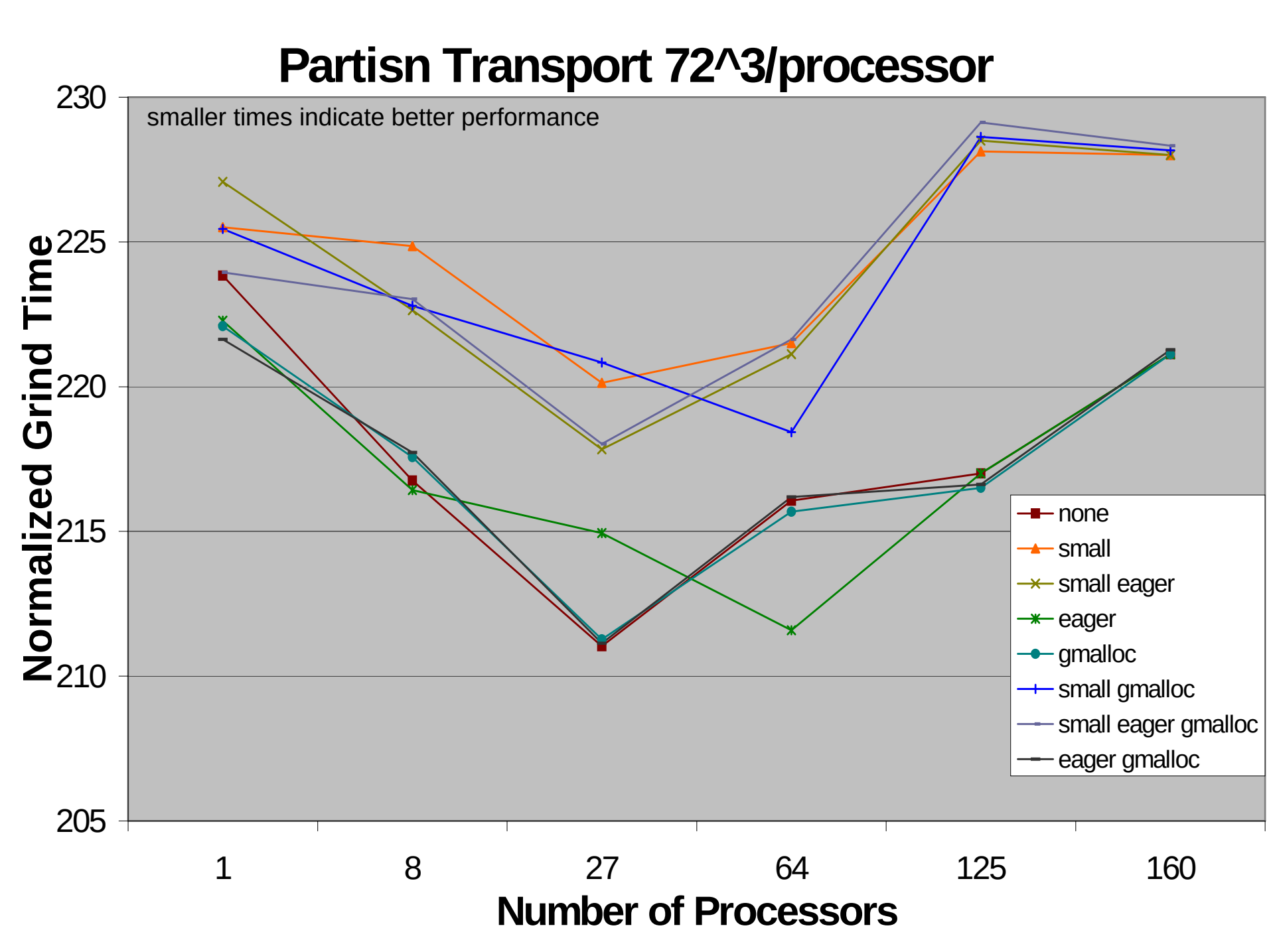
time = 0.3 ms

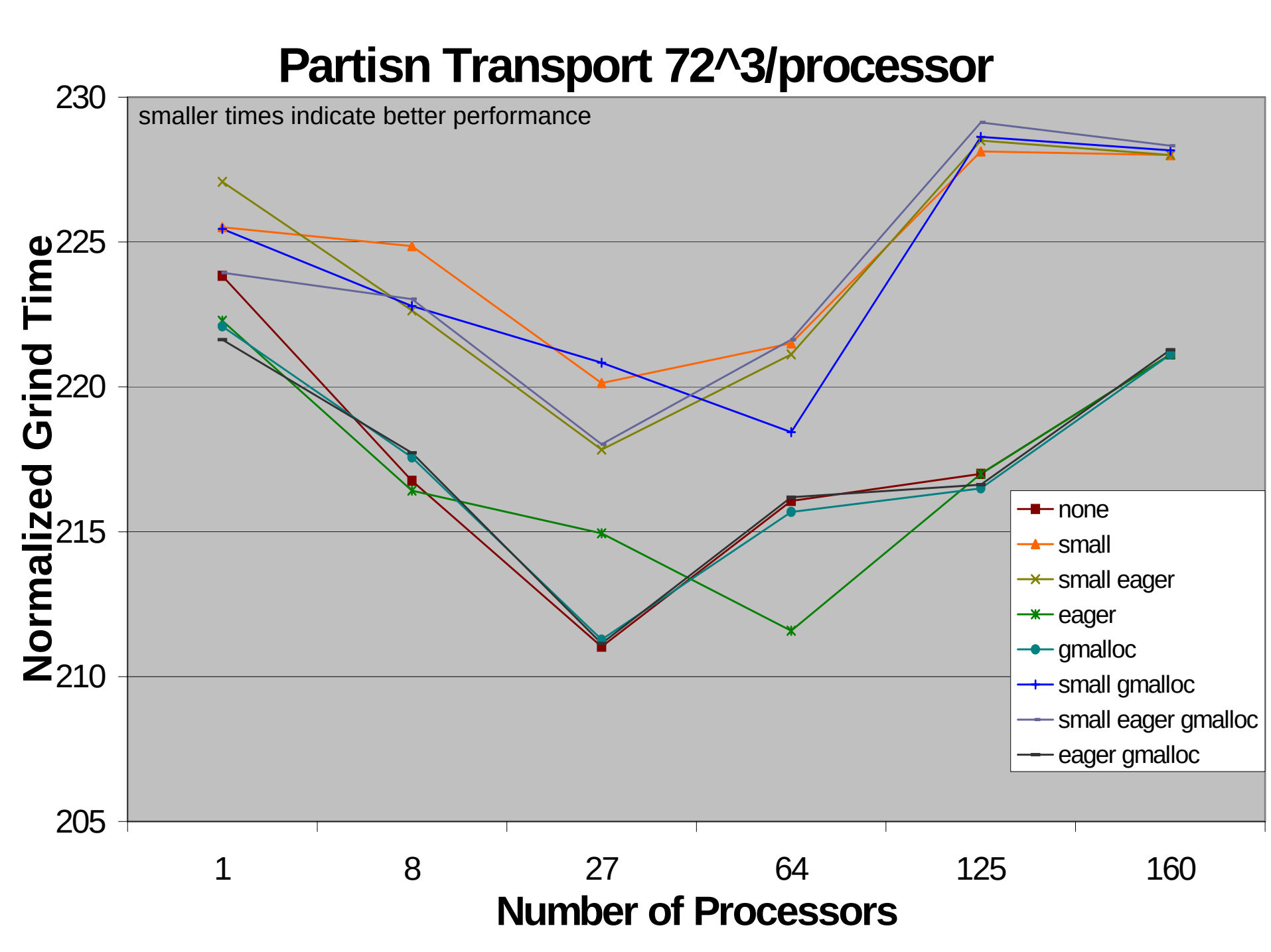
Sandia National Laboratories


CTH 7.1 - shaped charge 90 x 216 x 90/proc
smaller times indicate better performance
Time per Timestep
18
17
16
15
14
13
12
11
10
2 cabinet test system
Upgraded
1
2
4
8
16
32
64
128
128
256
512
1024
2048
Number of Processors
none
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc

Partisn

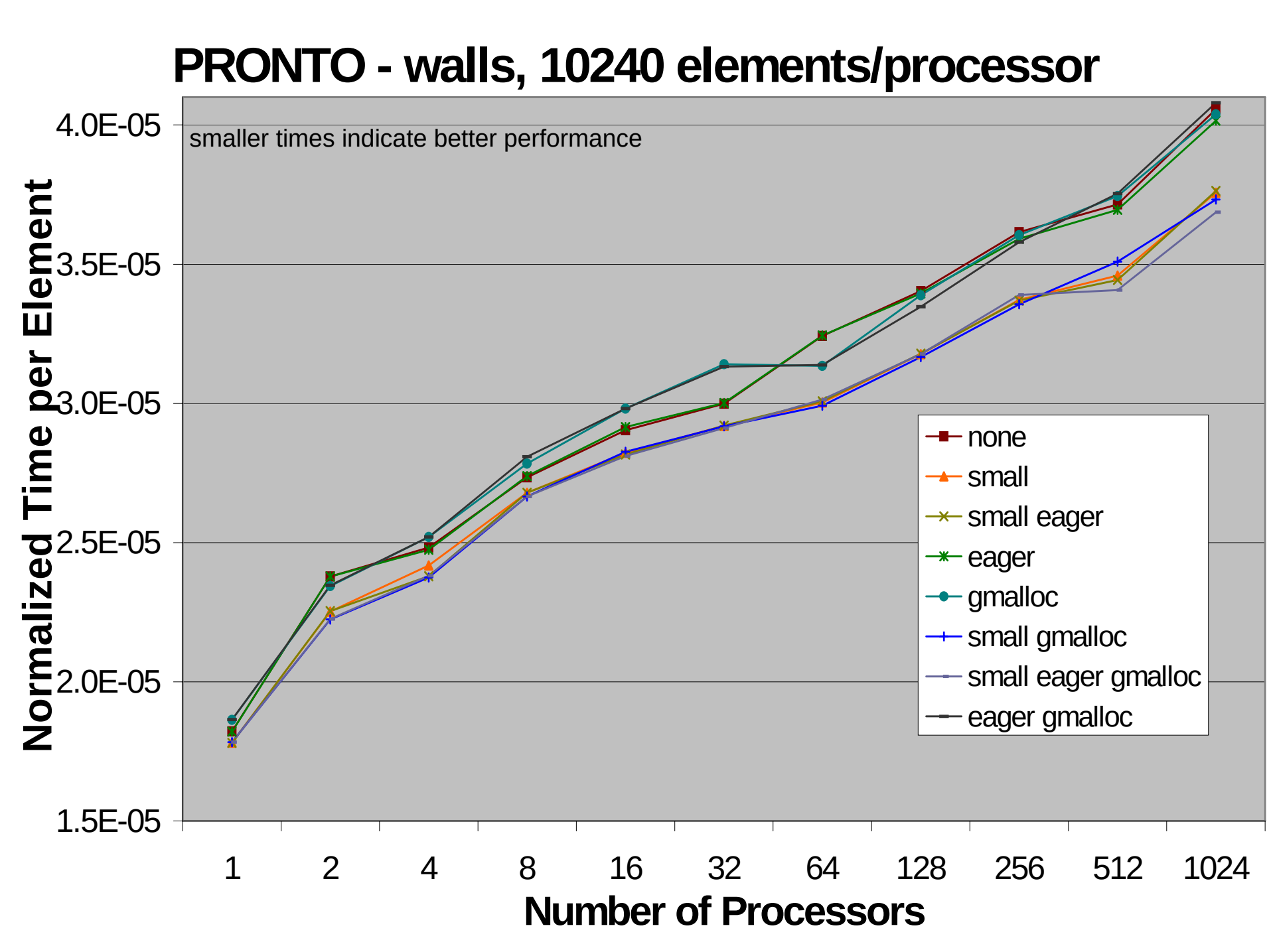
- LANL code that solves the Boltzmann transport equation
- Has a transport and diffusion phase
- SN timing problem with 72^3 cells per processor
- Run only on 2 cabinet test system
- Code is mostly FORTRAN


Partisn Transport 72^3/processor
smaller times indicate better performance
Normalized Grind Time
230
225
220
215
210
205
1
8
27
64
125
160
Number of Processors
none
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc


Partisn Transport 72^3/processor
smaller times indicate better performance
Normalized Grind Time
230
225
220
215
210
205
1
8
27
64
125
160
Number of Processors
none
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc

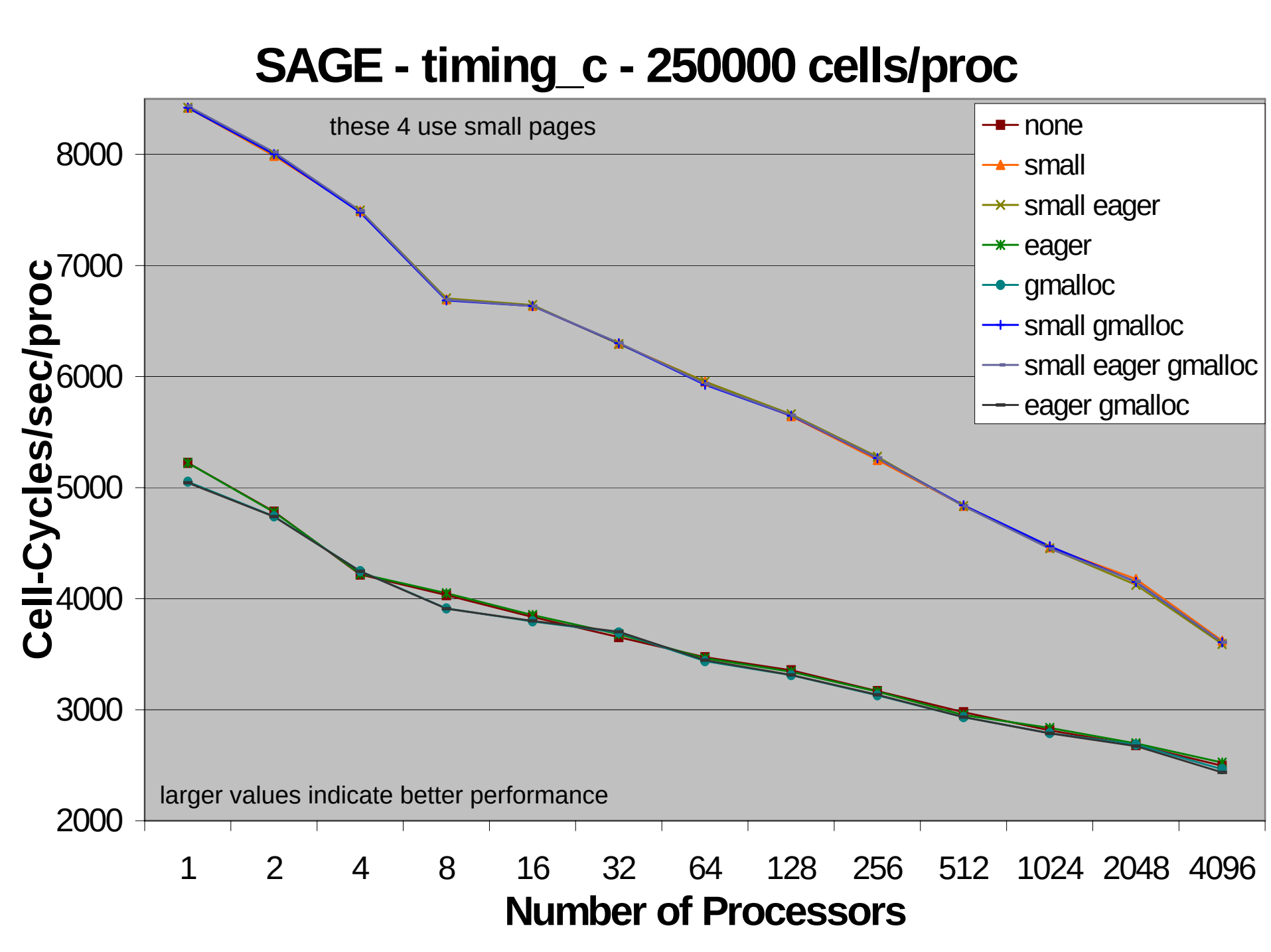
PRONTO

- Structural mechanics code with contact algorithm
- Walls problem - two sets of two brick walls colliding
- 10240 elements per processor
- Size is such that each brick is contained on a processor
  - Eliminates communication for the finite element portion of the calculation
  - All communication for contact portion
- Code is FORTRAN 90 with C for contact communication


PRONTO - walls, 10240 elements/processor
smaller times indicate better performance
Normalized Time per Element
4.0E-05
3.5E-05
3.0E-05
2.5E-05
2.0E-05
1.5E-05
1
2
4
8
16
32
64
128
256
512
1024
Number of Processors
none
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc

SAGE

- LANL Eulerian Hydrocode with AMR (Adaptive Mesh Refinement)
- timing_c problem
  - has adaptation and heat conduction
  - 250000 cells per processor
- Code is mostly FORTRAN 90


SAGE - timing_c - 250000 cells/proc
these 4 use small pages
none
small
small eager
eager
gmalloc
small gmalloc
small eager gmalloc
eager gmalloc
Cell-Cycles/sec/proc
8000
7000
6000
5000
4000
3000
2000
larger values indicate better performance
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
Number of Processors

Summary

- No set of options is always best
- Results from benchmarks do not necessarily translate to codes
- Small pages generally helps and can help significantly
- The other options have small effects for the codes that were tested

Future Work

- Profile codes and benchmarks to understand when each of the options helps
- Run test with a C++ code
- If accepted, an update of this work with these additions will be presented at CUG 07

Sandia National Laboratories